Software Engineering and Human Interface

CSET 3600 – 002

~~Team 4~~

Team Laser Explosion

Ryan Rapini
Adam Servis
Edward Verhovitz
Allyn Cheney

PROJECT PROPOSAL

FRIDAY, SEPTEMBER 21, 2012

Our group has decided to implement a battleship game for the course project. The battleship game will have options for single player vs. a computer or for two players to go head to head in an *exhilarating and intense* multiplayer battle of awesome. Players will take turns firing missiles at each other's immobile fleet of military vessels in what is one of the most visceral war-simulators ever created. There will of course be some interesting features that we will implement. Neither player will be able to see the other player's board of course because that would pretty much defeat the purpose of Battleship. The AI will have several difficulty levels ranging from Battleship Grand Master to bumbling oaf, giving singleplayer games a bit of depth and excitement as well. The game will also log all plays made by both players and allow for them to be played back later. When one person wins we will probably make the game sing them a song or show them a funny picture or something, too.

On the programming side, we plan to use Python as our language of choice. Python is versatile and powerful, and has some of the best documentation around in addition to an abundance of open-source libraries we can use to implement various features. Also the mascot is a snake and I really like snakes. Python allows us to program in a very flexible manner since it bundles its own live interpreter and compiles at runtime. This will allow for easy testing, and version control is trivial when no one has to worry about build environments or handling binaries. Python is also cross platform, which is nice, as it will allow us to program on whatever computers we have already. All of the code will be Object Oriented for Great Justice, using only the most rigid and proper of programming practices.

We plan to implement this as a GUI application with some of the most awesome graphics we can find. Adam Servis has already volunteered to handle all of our graphic design. What a nice guy. Hopefully we can be approved for this.

Ed is in charge of handling the main game mechanics, since he has already told me he is an avid Battleship enthusiast. He will be handling the rules of the game, i.e. what ships can be placed where and how to tell if you have hit or sunk a ship. This is his passion, we'll leave him to it. I will be in charge of the multiplayer aspect of the game, and the netcode. I like to think my vast knowledge of online games will make me a natural for programming networking infrastructure. I also read a book on networking once, so that should come in handy. Allyn will be in charge of the game's AI for single player, in all of its varying

complexities. He will also try to iron out any other bugs we come across along the way. He will also be in charge of composing the game's musical score and Original Sound Track.

Beyond this, we don't have a lot planned, but I'm sure we will make an awesome project!

USE CASES

October 12, 2012

Our battleship project will use a wide variety of use cases. The game will start with a main menu, from which the user will have the choice of either StartGameVsAI or StartGameVsPlayer. Regardless of which choices is picked, the program will bring them to a game board via the SetupGameBoard use case, which will then bring each user to the UserSetsShips use case. From there, the game will run via the RunGame use case, periodically CheckingForWinner and CheckForHit and CheckForSunkShip

Use Case: StartGameVsAI

Actors: Game player

Goal: To start a single player game vs. computer

Preconditions: Game is at menu

Trigger: User decides to play game vs. computer

Scenario:

1. Game menu is displayed
2. User clicks start game vs. computer button
3. System starts game vs. computer

Exceptions:

a. n/a

Use Case: StartGameVsPlayer

Actors: Game player 1 and 2

Goal: To start a networked game for 2 players

Preconditions: Game is at menu

Trigger: User decides to play game vs. another player

Scenario:

1. Game menu is displayed
2. User clicks start network game button
3. System goes to menu asking for network IP of second player
4. Player enters an IP
5. Game connects users

Exceptions:

- User doesn't enter IP for second player
- User doesn't enter valid IP for second player

Use Case: SetupGameBoard

Actors: System

Goal: To setup the game board

Preconditions: Game loading

Trigger: User has picked either game vs. AI or another player

Scenario:

1. Game is loading
2. Game sets up arrays to be used as game board
3. Game goes to game board

Exceptions:

- n/a

Use Case: UserSetsShips

Actors: Game player

Goal: To have user set ships up on their board

Preconditions: Game is loaded

Trigger: Game board has loaded

Scenario:

1. Game board is setup and game loaded
2. User places ships on their board

Exceptions:

- User fails to place all ships on the board, a message should popup informing them to place all their ships.
- User places ships improperly (overlapping) or off the board.

Use Case: RunGame

Actors: System

Goal: To start playing the game

Preconditions: Game is loaded and ships are placed

Trigger: User has clicked button after placing all their ships

Scenario:

1. Game is loaded and ships are placed
2. Randomly one player is prompted to start the game
3. Users keep picking locations one at a time until one person's ships are sunk

Exceptions:

- User picks a location they have already picked
- User picks an invalid location
- User leaves the game unexpectedly

Use Case: CheckforWinner

Actors: System

Goal: To check for game winner

Preconditions: Game is loaded, ships are placed and game is running

Trigger: User checks for a spot looking for a hit or miss

Scenario:

1. Game is running
2. User clicks a spot looking for a hit
3. Afterwards game checks to see if a user has won

Exceptions:

- n/a

Use Case: CheckforHit

Actors: System

Goal: To check if user hit a ship

Preconditions: Game is running

Trigger: User checks a spot

Scenario:

1. Game is running
2. User clicks a spot looking for a hit or miss
3. System returns either hit or miss

Exceptions:

- User clicks a spot they have already checked

Use Case: CheckforSunkShip

Actors: System

Goal: To check if user sunk a ship

Preconditions: Game is running

Trigger: User checks a spot

Scenario:

1. Game is running
2. User clicks a spot looking for a hit or miss
3. System informs user if they sunk a ship

Exceptions:

- n/a

Use Case: CheckforMiss

Actors: System

Goal: To check if user missed a ship

Preconditions: Game is running

Trigger: User hits empty field of play

Scenario:

1. Game is running
2. User clicks a spot looking to hit a ship
3. System informs user of a miss

Exceptions:

- n/a